Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 28 (2014) 95 – 102
Procedia Computer Science

Conference on Systems Engineering Research (CSER 2014)

Eds.: Azad M. Madni, University of Southern California; Barry Boehm, University of Southern California; Michael Sievers, Jet Propulsion Laboratory; Marilee Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 21-22, 2014

Exploiting Architectural Communities in Early Life Cycle Cost Estimation

Matthew Dabkowski[a]*, Ricardo Valerdi[a], John Farr[b]

[a]*University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721-0012, USA*
[b]*United States Military Academy, Building 752 - Mahan Hall, West Point, NY 10996-1779, USA*

Abstract

System architectures evolve over time. Accordingly, the dynamic properties of architectures reflect how systems respond to change, and this response ultimately impacts cost. In prior work we make an explicit connection between the architectural diagrams of Model-Based Systems Engineering (MBSE), parametric cost estimation, and network science. Specifically, by treating the DoD Architecture Framework (DoDAF) Systems View 3 (SV3) as an adjacency matrix, we assess how the addition of a new subsystem to an immature architecture might grow the existing network. With the subsequent application of parametric cost modeling, we translate anticipated growth into expected cost, thereby quantifying the impact of change. This paper refines that approach. In particular, by using the Girvan-Newman algorithm, the SV3 is initially divided into groups of subsystems such that the number of interfaces is dense within and sparse between groups. Based on this division into "architectural communities" and the prevalence of bridging ties, interfaces generated by the addition of a new subsystem can be faithfully integrated into the existing architecture, adding validity to our growth mechanism. This procedure is illustrated in detail with an example that highlights the importance of this refinement, and it is incorporated within a Monte Carlo simulation that allows the distribution of future costs to be estimated and assessed.

* Corresponding author. Tel.: +1-520-248-6130; fax: +1-520-621-6555.
E-mail address: mfd1@email.arizona.edu.

doi:10.1016/j.procs.2014.03.013

Keywords: Model Based Systems Engineering; DoD Architecture Framework; COSYSMO; network science; community detection

1. Introduction

Since its mathematical formalization in 1993,[1] MBSE has experienced prodigious methodological growth[2] and shown tremendous practical promise.[3] Concurrent with the MBSE revolution, the capabilities and acceptance of commercial parametric cost models have increased dramatically.[4] With the complexity of systems increasing[5] and cost overruns seeming the norm,[6,7] both MBSE and parametric cost modeling appear poised for continued growth, and their marriage seems natural. Nonetheless, architectural analysis and cost estimation are often disconnected, especially early in the life cycle when uncertainty is high and confidence is low. Unfortunately and to the detriment of decision makers, this is also when a system's architecture is most pliable.[8]

2. Linking MBSE, Parametric Cost Modeling, and Network Science

Based on the above, the integration of MBSE and parametric cost modeling represents a fertile area for study, and our 2012 paper titled "Network Science Enabled Cost Estimation in Support of MBSE" makes this connection explicit.[9] In particular, we demonstrate how DoDAF's SV3 can serve as an input for the Constructive Systems Engineering Cost Model (COSYSMO). Additionally, by abstracting the SV3 as a network, we subsequently use the Barabási–Albert preferential attachment (PA) model to simulate growth in the architecture, and, ultimately, estimate the cost of change. While the interested reader should consult our prior work for a full description, a brief accounting is necessary to motivate the remainder of this paper.

2.1. Illustrative Example

In order to provide context for our existing methodology, consider the hypothetical system in Figure 1 below. Consisting of 20 subsystems (labeled **A** through **T**) and 47 interfaces, our system is assumed to be under development yet mature enough for interface complexity to be assessed. Moreover, without loss of generality, assume there are "200 easy, 200 nominal, and 50 difficult requirements, as well as 5 difficult critical algorithms."[10]

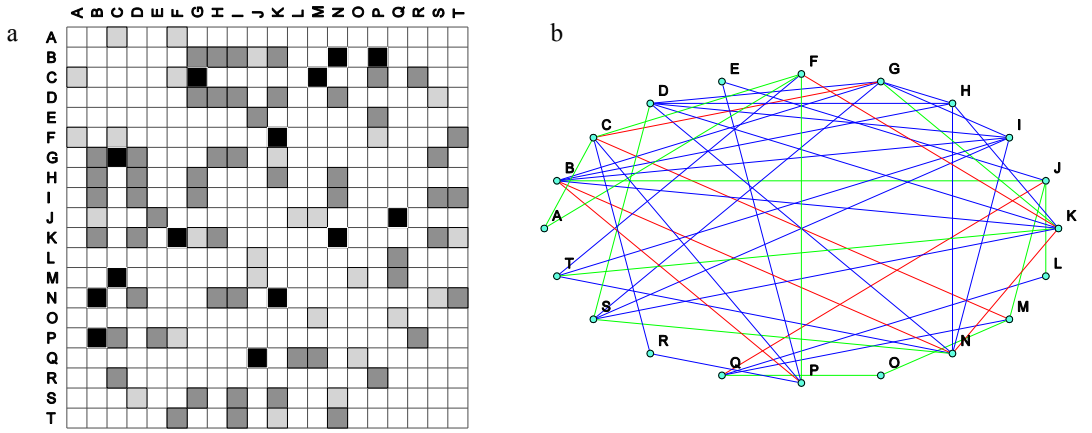


Fig. 1. (a) This matrix represents the weighted SV3 (adjacency matrix), $\boldsymbol{W}$, for our hypothetical system, where the shading of the cells indicates interface complexity (light gray ⇒ easy, medium gray ⇒ nominal, black ⇒ difficult). (b) The same system represented as an undirected, weighted graph, where the edge weights (w_{ij}) are indicated by the color of the edge (green ⇒ w_{ij} = 1.1, blue ⇒ w_{ij} = 2.8, red ⇒ w_{ij} = 6.3).

Based on this information, an estimate of the systems engineering effort (in person months (PM)) required to bring our system to fruition can be generated from COSYSMO's cost estimating relationship (CER), specifically:

$$PM_{NS} = A \cdot \underbrace{\left(\sum_{i \in \{e,n,d\}} \sum_{k=1}^{4} w_{i_k} \Phi_{i_k} \right)^{E}}_{\text{size}} \cdot \underbrace{\prod_{j=1}^{14} EM_j}_{\text{effort}} \tag{1}$$

where . . .

PM_{NS} = system engineering effort (nominal schedule),
A = calibration constant derived from historical project data (assume as 0.25),
w_{ik} = weight for the i^{th} complexity level of the k^{th} size driver ($i \in \{e$ (easy), n (nominal), d (difficult)$\}$),
Φ_{ik} = quantity of the k^{th} size driver with complexity level i ($k \in \{1$ (requirements), 2 (interfaces), 3 (algorithms) and 4 (operational scenarios)$\}$),
E = diseconomies of scale constant (assume as 1.06), and
EM_j = systems engineering effort multiplier for the j^{th} cost driver (assume product is 0.89).[11]

Using the w_{ik} contained in our prior work[12] and solving for PM_{NS}, we conclude that 245.27 PM are required. If we further assume that each PM costs $20,000, this equates to $4,905,400.

2.2. Existing Methodology

As with our 2012 paper, suppose we are interested in estimating the effort required to incorporate an additional subsystem (**U**) into the architecture without knowing its purpose or function. In light of COSYSMO's CER, this ultimately forces us to estimate the number of interfaces (by complexity level) **U** will generate. More granularly, we need to answer three questions: (a) How many subsystems should **U** connect to (degree)?; (b) Given **U** connects to d subsystems, which d subsystems should it connect to (adjacency)?; and (c) Given **U** connects to a specific set of d subsystems, what should the complexity of these interfaces be (weights)?.

Starting with (a), we elected to employ a "rich-by-birth" effect, where we view the degree of **U** ($D_{\mathbf{U}}$) as a random variable with a probability mass function (pmf) equal to the observed degree distribution of the existing system. For (b), we utilized the PA model to incorporate a "rich-get-richer" effect, where highly connected subsystems are more likely to interface with **U**. Lastly, for (c) we modeled the complexity of the interface between **U** and subsystem i ($w_{i\mathbf{U}}$) as a conditional random variable, where the pmf for $w_{i\mathbf{U}}$ equates to the observed interface complexity distribution of subsystem i.[13]

Armed with a method to estimate the interfaces **U** will generate, we implemented this procedure in the statistical software R, and our corresponding pseudo-code is as follows:

For a specified number of iterations (n) . . .
(1) Initialize the system as the current system,
(2) Generate a realization for $D_{\mathbf{U}}$ (d); this is the number of subsystems **U** will attach to,
(3) Connect **U** to d subsystems of the current system using the PA model,
(4) For each interface established in (2), assign complexity ($w_{i\mathbf{U}}$),
(5) Estimate the cost for the augmented system using COSYSMO (PM_{NS}*),
(6) Calculate the additional cost of adding subsystem **U** (PM_{NS}* − PM_{NS}), and
(7) Store results and return to (1).[13]

Setting n equal to 10,000 and running the simulation, we find that the expected cost of adding subsystem **U** is $86,160, and it will likely not exceed its maximum observed value of $280,200.

2.3. A Thought Experiment

In light of the above example, a natural question is "How reliable is the existing methodology?" Put another way, "Are there situations where its robustness is questionable?" Consider the system depicted in Figure 2 below.

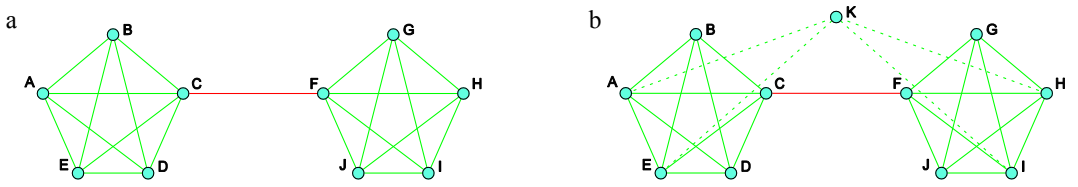


Fig. 2. (a) A hypothetical system consisting of 10 subsystems and 21 interfaces, where interface complexity is indicated by the color of the edge (green ⇒ easy, red ⇒ difficult). (b) A realization of the same system following the addition of subsystem **K**.

As seen in Panel (a) of Figure 2, the system clearly consists of two clusters (or communities) of subsystems, namely, community 1 = {**A**, **B**, **C**, **D**, **E**} and community 2 = {**F**, **G**, **H, I**, **J**}. Moreover, each community is not only complete (implying each subsystem in community *i* directly interfaces to every other subsystem in community *i*) but also entirely composed of easy interfaces. Finally, the two communities are linked by a single interface (edge **C**-**F**), and this *bridging* interface is categorized as difficult.

Now, suppose we perform a single iteration of our existing methodology on the system depicted in Panel (a) of Figure 2, and it returns the realization seen in Panel (b). In this instance, our new subsystem **K** connects to **A** and **E** in community 1 and **H** and **I** in community 2, and these interfaces are rated as easy. While this is plausible based on the existing methodology, is it realistic? We argue "probably not." Specifically, if we assume that the existing architecture foretells the future architecture (an assumption implicit in the existing methodology), then the realization in Panel (b) is spurious. After all, the existing architecture suggests (1) the system consists of two well defined communities; (2) bridging interfaces are rare; and (3) intercommunity interfaces are difficult. The realization in Panel (b) blatantly defies all three. Even worse, the likelihood of realizing these violations is high. While troubling for our methodology, network science offers us a computational fix, namely, community detection.

3. Community Detection

Within network science community detection represents a class of clustering algorithms that find "naturally occurring groups in a network regardless of their number or size . . . [such that the network is separated] . . . into groups of vertices that have few connections between them."[14] For our purposes, community detection provides us with a mechanism to identify a system's "architectural communities" – groups of subsystems where the number of interfaces is dense within and sparse between groups. Although there are many heuristics to effectively and efficiently detect communities in a network, the well-known Girvan-Newman algorithm is appealing for several reasons. First and foremost, the algorithm has performed well in a variety of contexts, to include community detection in man-made systems.[15,16,17] Next, unlike other heuristics that return only the "best" partitioning, Girvan-Newman provides a dendogram (or tree) displaying multiple possibilities for a network's community structure.[18] Lastly, despite its computational complexity, it is intuitive and simple enough to explain in a few figures, a characteristic we feel increases not only its face validity but also its salability.

3.1. The Girvan-Newman Algorithm

At its core, the Girvan-Newman algorithm is based on the idea of *edge betweenness*. Loosely defined, an edge's betweenness is the number of geodesic or shortest paths in the network that contain the edge.[18] Intuitively, edges with high edge betweenness bridge communities of vertices (or subsystems). Drawing on this idea, a sketch of the Girvan-Newman algorithm is as follows: (1) Given network Ω of size *n*, calculate the edge betweenness for each

edge in Ω; (2) Delete the edge with the highest edge betweenness from Ω, yielding subgraph Ω′; (3) If Ω′ has 0 edges, terminate; otherwise, set Ω as Ω′ and return to (1).[19] At termination, the algorithm produces a dendogram with n leaves, and cutting the tree at different levels of the hierarchy produces different community realizations.[18] Ultimately, the realization that maximizes *modularity* (to be defined below) is selected.[18]

To make this description more concrete, a simple example is warranted. Accordingly, consider the graph (Ω) given in Panel (a) of Figure 3 below. In Panel (b), we calculate the edge betweenness (ω) for each edge in Ω. Using edge **B-C** as an example, note that (1) **B-C** is the shortest path between **B** and **C** ($\omega_{\mathbf{B\text{-}C}} = 1$), (2) **B-C** is on the unique shortest path between **B** and **F** (**B-C-F** ⇒ $\omega_{\mathbf{B\text{-}C}} = 2$), and (3) **B-C** is on one of two shortest paths between **B** and **E** (**B-A-D-E** and **B-C-F-E** ⇒ $\omega_{\mathbf{B\text{-}C}} = 2.5$). Similar calculations are made for the remaining edges in Ω, and edge **A-D** has the highest ω. As such, we delete this edge from Ω, and the resulting graph (Ω′) has at least one edge. Thus, Ω is set as Ω′, and ω is calculated again for each edge in Ω. As Panel (c) shows, edge **C-F** now has the highest ω, and it is deleted from Ω. In Panel (d), we note that Ω is now fragmented into two communities.

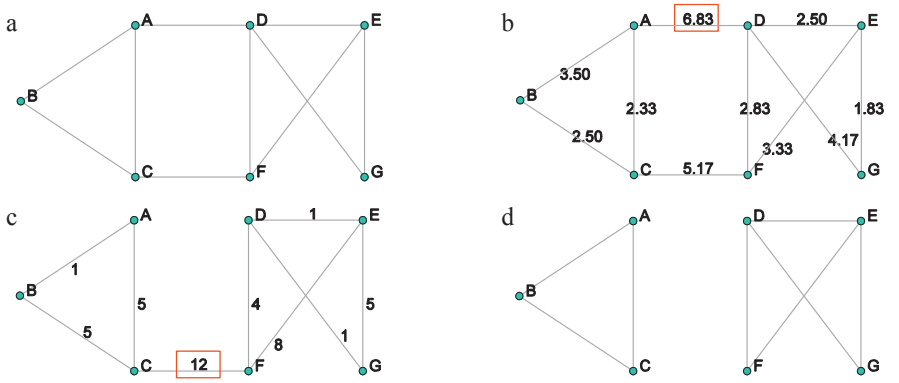


Fig. 3. Illustration of edge betweenness, edge deletion, and fragmentation, where panel descriptions are provided in the accompanying text.

The natural question is "How 'good' is this partitioning?" To address this, we introduce the metric known as modularity (Q) which is given formulaically in Equation (2) below, where n is the number of communities; e_{ij} is the fraction of edges between communities i and j (and represents the entry in row i and column j of matrix **e**); and a_i is the sum of row i of matrix **e**.[20] Moreover, as Girvan and Newman note: "If the number of within-community edges is no better than random, we will get $Q = 0$. Values approaching $Q = 1$, which is the maximum, indicate networks with strong community structure."[20]

$$Q = \sum_{i=1}^{n} \left(e_{ii} - a_i^2 \right) \tag{2}$$

As seen in Figure 4 below, the modularity for the community structure given in Panel (d) of Figure 3 is 0.28.

e	1	2
1	0.3	0.1
2	0.1	0.5

e_{ii}	a_i	$e_{ii} - a_i^2$
0.3	0.4	0.14
0.5	0.6	0.14
	Q	0.28

Fig. 4. Modularity calculation for the division of Ω into {**A**,**B**,**C**} and {**D**,**E**,**F**,**G**}. Specifically, using the full graph, e_{ij} is calculated between communities i and j (for all i, j) producing matrix **e**. For example, e_{22} is 0.5, indicating 50% of all the edges contained in Ω reside within the second community {**D**,**E**,**F**,**G**}. With **e** in hand, we subsequently apply Equation (2) to find Q.

This process continues in a similar manner until Ω has no edges remaining, and, at termination, the dendogram has seven leaves with six potential community structures. As Figure 5 shows, our initial fragmentation maximizes modularity at 0.280; thus, we conclude Ω contains two communities, namely, {**A**,**B**,**C**} and {**D**,**E**,**F**,**G**}.

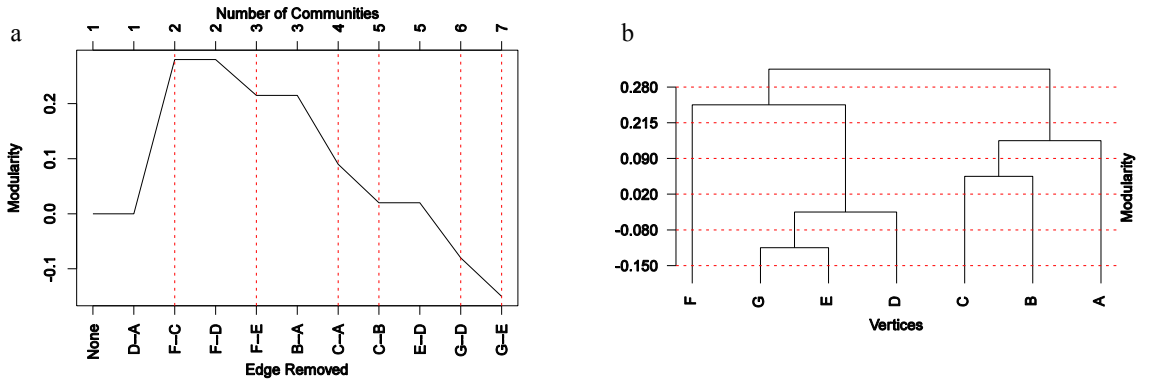


Fig. 5. (a) Modularity versus edge removed, where the removal sequence is given from left to right on the horizontal axis. As this graph shows, modularity is maximized at 0.280 after the removal of edge F-C. (b) Dendogram displaying the modularity of different community realizations.

3.2. Identifying Architectural Communities

Armed with an understanding of Girvan-Newman, we applied it to the hypothetical system seen in Figure 1. Specifically, using the *edge.betweenness.community* function from R's *igraph* package,[21] the modularity maximizing partition split the system into three communities, namely: community 1 = {**Q**, **O**, **E**, **L**, **M**}, community 2 = {**N**, **G**, **H**, **D**, **I**, **K**, **T**, **B**, **S**}, and community 3 = {**F**, **P**, **A**, **R**, **C**}. While the modularity value of 0.313 is distant from the theoretical maximum, it is practically significant.[22] Furthermore, when the subsystems are permuted by their community membership (Figure 6), the partition's veracity is quite apparent.

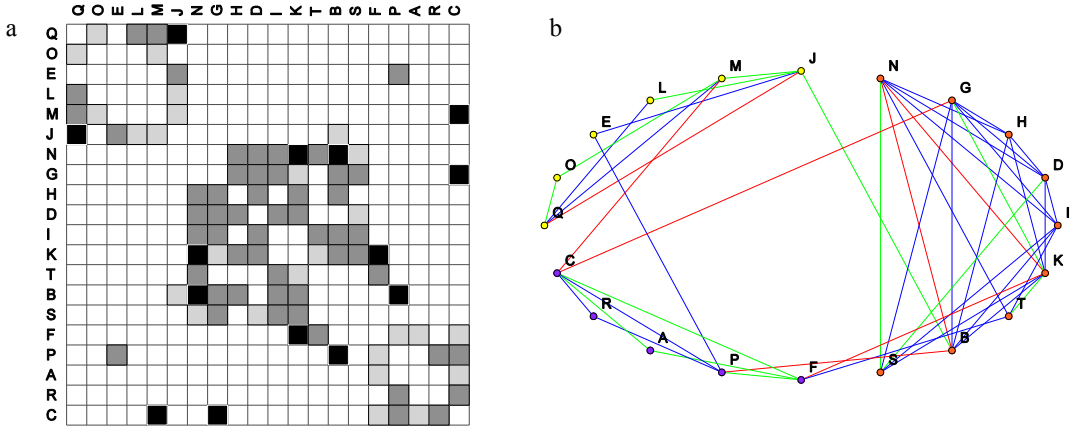


Fig. 6. (a) An isomorphic representation of $\boldsymbol{W}$ (following permutation). (b) The same system displayed as an undirected, weighted graph, where vertex (subsystem) color indicates community membership (yellow ⇒ community 1, orange ⇒ community 2, purple ⇒ community 3).

3.3. Exploiting Architectural Communities

As seen visually in Figure 6 above and numerically in Table 1 below, the density[23] (Δ) of intracommunity interfaces is much greater than intercommunity interfaces. Moreover, interface complexity varies considerably based on not only community membership but also whether an interface links intra or intercommunity subsystems.

Table 1. Intra and Intercommunity Density.

Community	Δ	Communities	Δ
1	0.5333	1 and 2	0.0095
2	0.6944	1 and 3	0.0364
3	0.7000	2 and 3	0.0440

In short, community membership matters, and we integrate it into the existing methodology as follows:

For a specified number of iterations . . .

Preprocessing

(1) Initialize the system as the current system,
(2) Use Girvan-Newman to identify architectural communities,
(3) Randomly assign **U** to community *j,*

Intracommunity Growth

(4) Generate a realization for $D_{\mathbf{U},\text{intra}}$ given **U** is assigned to community *j* (d_{intra}),
(5) Connect **U** to d_{intra} subsystems inside community *j* using the PA model,
(6) For each interface established in (5), assign complexity ($w_{i\mathbf{U},\text{intra}}$),

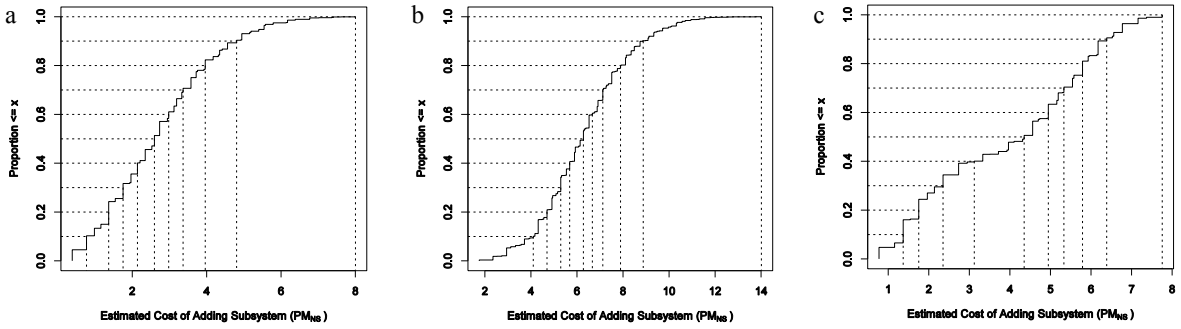
Intercommunity Growth

(7) Generate a realization for $D_{\mathbf{U},\text{inter}}$ given **U** is assigned to community *j* (d_{inter}),
(8) Connect **U** to d_{inter} communities using the PA model,
(9) For each interface established in (8), assign complexity ($w_{i\mathbf{U},\text{inter}}$),

Cost Estimation

(10) Estimate the cost for the augmented system using COSYSMO (PM_{NS}*),
(11) Calculate the additional cost of adding subsystem **U** (PM_{NS}* − PM_{NS}), and
(12) Store results and return to (3).

As seen above, by (1) adding community detection as a preprocessing step and (2) subsequently exploiting it via intra and intercommunity growth, our revised methodology more fully utilizes the information available in the SV3. Moreover, as seen in Figure 7 below, by integrating community membership directly into our methodology, we are able to generate a more detailed estimate for the cost of adding subsystem **U** to the existing architecture. For example, if **U** is added to architectural community 2, our expected (or mean) cost is \$126,520 or \$40,360 (46.8%) more than our previous estimate. On the other hand, if **U** is added to architectural community 1, our expected cost is \$31,420 (36.5%) less.



- 95% CI is (2.68, 2.78) in PM_{NS}
- "Best guess" is \$54,740
- Cost should not exceed \$160,020

- 95% CI is (6.25, 6.39) in PM_{NS}
- "Best guess" is \$126,520
- Cost should not exceed \$280,240

- 95% CI is (3.86, 4.00) in PM_{NS}
- "Best guess" is \$78,720
- Cost should not exceed \$155,120

Fig. 7. Empirical cumulative distribution functions and statistics for the cost of adding subsystem **U** to communities (a) 1, (b) 2, and (c) 3.

4. Limitations

As with our prior work, our revised methodology has several limitations. First, by using a system's existing architecture to estimate its future architecture, we fail to account for revolutionary change. For example, recall Panel (b) in Figure 2, where subsystem **K** bridges the existing architectural communities with easy interfaces. In this case, **K** is serving the role of a hub, a revolutionary change that could ultimately eliminate the need for the difficult interface between subsystems **C** and **F**. Nonetheless, the revised methodology was designed to eliminate this type of realization, and, thus, addresses evolutionary versus revolutionary change. Additionally, one might argue that technological interfaces are not random; they are deliberately engineered based on requirements. While we wholeheartedly concur with this sentiment, we argue that detailed interfaces are engineered after the requirements mature, and this is an early life cycle estimate. Lastly, we acknowledge that the revised (as well as the existing) methodology is not "estimation ready" in its current form, as COSYSMO's parameters must be calibrated first. Although this may seem like a distinct disadvantage, we actually view this wrinkle as a safeguard – one that ensures our methodology is well understood (and fit for purpose) before it is implemented.

5. Future Work

Without question, the immediate focus of our future work involves validating the methodology using real-world systems. After making appropriate adjustments, we would like to integrate our approach into commercially available software, thereby providing practitioners with a means to increase the accuracy and fidelity of their analyses early in the life cycle. More generally, we hope this work amplifies the utility of and the opportunity for further research on the integration of network science and systems engineering.

References

1. Wymore AW. *Model-Based Systems Engineering*. Boca Raton, FL: CRC Press; 1993.
2. INCOSE MBSE Initiative. Survey of Model-Based Systems Engineering (MBSE) Methodologies. Seattle, WA: INCOSE; 2008.
3. NDIA Systems Engineering Division. Final Report of the Model Based Engineering (MBE) Subcommittee. Arlington, VA: NDIA; 2011.
4. International Society of Parametric Analysts (ISPA). Parametric Estimating Handbook, 4th ed. Vienna, VA: IPSA; 2008.
5. INCOSE Technical Operations. Systems Engineering Vision 2020, version 2.03, INCOSE-TP-2004-004-02.5. Seattle, WA: INCOSE; 2007.
6. GAO. DOD COST OVERRUNS: Trends in Nunn-McCurdy Breaches and Tools to Manage Weapon Systems Acquisition Costs, GAO-10-106; 2011.
7. Flyvbjerg B, Skamris Holm MK, Buhl SL. How common and how large are cost overruns in transport infrastructure projects? *Transport Reviews* 2003; **23** 1:71-88.
8. Blanchard B, Fabrycky W. *Systems engineering and analysis*. 3rd ed. Upper Saddle River, NJ: Prentice Hall; 1998. p. 37.
9. Dabkowski M, Reidy B, Estrada J, Valerdi R. Network Science Enabled Cost Estimation in Support of Model-Based Systems Engineering. *Procedia Computer Science* 2013; **16**: 89-97.
10. Ibid. p. 92.
11. Valerdi R. The Constructive Systems Engineering Cost Model (COSYSMO). PhD Dissertation. Los Angeles, CA: University of Southern California; 2005. p. 31-38.
12. Dabkowski M, Reidy B, Estrada J, Valerdi R. Network Science Enabled Cost Estimation. p. 91-92.
13. Ibid. p. 94-95.
14. Newman MEJ. *Networks: An Introduction*. New York: Oxford University Press; 2010. p. 371.
15. Newman MEJ, Girvan M. Finding and evaluating community structure in networks. *Physical Review E* 2004; **69** 026113:1-15.
16. Girvan M, Newman MEJ. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 2002; **99** 12:7821-26.
17. Bogojeska A. Topological and structural analysis of the electric power grid of Southeast Europe. *Proceedings of the 3rd International Conference on Information, Society, and Technology* 2013; 12-16.
18. Newman MEJ. *Networks*. p. 382-385.
19. Girvan M, Newman MEJ. Community structure in social and biological networks. p. 7823.
20. Newman MEJ, Girvan M. Finding and evaluating community structure in networks. p.7.
21. Csardi G, Nepusz T. The igraph software package for complex network research. *InterJournal* 2006; **Complex Systems**:1695; http://igraph.sf.net.
22. Newman MEJ, Girvan M. Finding and evaluating community structure in networks. p.7.
23. Wasserman S, Faust K. *Social Network Analysis: Methods and Applications*. New York: Cambridge University Press; 2009. p. 101.